Hochschule Aalen
Fakultät Elektronik und Informatik
Studienbereich Informatik

Advanced Programming with MOSTflexiPL

Lecture in Wintersemester 2025/2026
Prof. Dr. habil. Christian Heinlein

5. Task Sheet (December 2, 2025)

Task 9: Lambda Parameters

Subtask 9.a)

Modify the operators for conjunction and disjunction from task 8.a in such a way, that the right operand is evaluated only if necessary, i. e., if its value is returned as the result value.

Modify the chained comparison operators from task 7 and 8.b also in such a way, that every operand is evaluated only if its value is needed to determine the result value, that means:

- The first and the second operand are always evaluated.
- Every subsequent operand is evaluated only, if the comparisons of all preceding operands have returned `true`.

Furthermore, every operand is evaluated at most once, even if its value is used two times.

Subtask 9.b)

Define an operator `if•then•{elseif•then•}[else•]end` for branches with the following meaning:

- The conditions, i. e., the operands after `if` and `elseif`, can have arbitrary (even different) types.
- All operands after `then` and possibly `else` must have the same type, however, which can also be arbitrary and which also constitutes the result type of the branch.
- At run time, the conditions are evaluated one after the other until one of them does not return nil, and then the value of the corresponding operand after `then` is returned.
- If all conditions return nil, the value of the operand after `else` is returned, if present, otherwise nil.
- Other operands are not evaluated.

Subtask 9.c)

Define an operator `(while|until|do)•{(while|until|do)•}end` for loops with the following meaning:

- All operands can have arbitrary (even different) types.
- At run time, one operand after the other is evaluated again and again until an operand after `while` returns nil or an operand after `until` does not return nil.
  The values of the operands after `do` are irrelevant.
- The result value of type `int` is the number of complete iterations, i. e., the number of iterations where all operands have been evaluated.

For example:

- Head-controlled loop with continuation condition:

  ```
  while ... do ... end
  ```

- Foot-controlled loop with termination condition:

  ```
  do ... until ... end
  ```

- Loop with multiple continuation and termination conditions at different places:

  ```
  do ... while ... do ... until ... do ... end
  ```

  Here, the following takes place in every iteration:
  - The first two operands are evaluated.
  - If the second returns nil, the loop is terminated.
  - Otherwise, the next two operands are evaluated.
  - If the second of them does not return nil, the loop is terminated.
  - Otherwise, the last operand is evaluated, and the next iteration is started.

- Infinite loop:

  ```
  do ... end
  ```

Task 10: Faculty and Power

a) Define a postfix operator •! that recursively computes the faculty of an `int` value. If the operand is not greater than or equal to 0 (i. e., less than 0 or unnatural), the result value shall be nil.

b) Define an infix operator `•^•` that efficiently computes powers, as described in the Wikipedia article “Exponentiation by squaring.”
Consider all possible special cases such as negative exponents (`x ^ k` shall be equivalent to `1 : (x ^ -k)` for `k < 0`) and unnatural values (in which case the result value shall be nil).
`x ^ 0` shall be equal to 1 for all natural values of `x` including 0.

c) Define the precedence and associativity of these operators as follows:
- The power operator is right-associative and binds stronger than the basic arithmetic operations including change sign. (Nevertheless, change sign as the right operand of power shall still be possible.)
- The faculty operator binds even stronger than the power operator.